UNCLASSIFIED

DEFENCE RESEARCH ESTABLISHMENT
CENTRE DE RECHERCHES POUR LA DÉFENSE
VALCARTIER, QUÉBEC

DREV - TN-2000- 50
Illimited Distribution/Distribution illimitée

DESCRIPTION OF THE DREV98 IMAGE FORMAT

by
M. P. Lévesque
January/janvier 2001

Approved by/approuvé par

Section head / Chef de section

11 Jan 2001
Date

SANS CLASSIFICATION

WARNING NOTICE

The information contained herein is proprietary to Her Majesty and is provided to the recipient on the understanding that it will be used for information and evaluation purposes only. Any commercial use, including use for manufacture, is prohibited. Release to third parties of this publication or of information contained herein is prohibited without the prior written consent of DND Canada.

© Her Majesty the Queen in Right of Canada as represented by the Minister of National Defence, 2001

ABSTRACT

The DREV98 image format was developed to conduct a number of experiments on the transfer of metadata between an auto-documented sensor and an auto-loading image database. These experiments allowed us to determine which metadata had to be stored in the image header. Moreover, this image format brings the c oncept that the header segments should contain meta-metadata, i.e. information that describes the header segment data (which itself describes the image) allowing a universal decoder to decode all unkno wn s egment. This technical note contains the description of the DREV98 image format and the new concept of the self-documented header segments.

RÉSUMÉ

On a mis au point le format d'image DREV98 pour effectuer un certain nombre d'expériences sur le transfert de méta-données entre un capteur autodocumenté et une base de données à chargement automatique. Ces expériences nous ont permis de déterminer quelles méta-données devraient être sauvegardées dans l'en-tête de l'image. De plus, ce format d'image amène le concept que les segments de l'en-tête pourraient contenir des méta-méta-données, i.e. de l'information qui décrit les données des segments (lesquelles decrivent elles-mêmes l'image), permettant à un décodeur universel de décoder n'importe quel segment inconnu. Cette note technique contient la description du format d'image DREV98 et du nouveau concept de segments auto-documentés.

This page is left blank

This page is left blank

TABLE OF CONTENTS

ABSTRACT/RÉSUMÉ i

EXECUTIVE SUMMARY v

ABBREVIATIONS vii

1.0 INTRODUCTION 1

2.0 THE HEADER STRUCTURE 3

3.0 THE MAIN HEADER 4

4.0 THE SECONDARY HEADER SEGMENTS 8

4.1 Why Adding Secondary Header Segments 8
4.2 The Generalization of the Header Segment 8
4.3 Description of the Structure of the DREV98 Secondary Header Segments 9

5.0 RECOMMENDATIONS FOR THE NEXT VERION 12

5.1 Bug in Fortran Encoding to C Decoding. 12
5.2 Identification of the Segments 12
5.3 Segment Format Descriptor 12

6.0 CONCLUSION 14

7.0 REFERENCES 15

FIGURES 1 and 2
TABLES I to IV

APPENDIX A 16

This page is left blank

This page is left blank

EXECUTIVE SUMMARY

Within the framework of the ACIDE (auto Context Image Database Exploitation) project, an ad-hoc image file format was developed. This image format allowed us to control the parameters of an experiment which consists in the communication of information between a hypothetical advance sensor able to auto-document its images and a database capable of loading automatically the image metadata.

This image format was not developed with the intention of being in competition with COTS image formats. It was rather developed to see which information should be packaged with the images (which the current formats do not allowed). This was useful 1) for our experiments and 2) to make suggestions to upgrade the current COTS image formats.

1.0 INTRODUCTION

This technical note contains information about the "DREV98" image header. "DREV98-a" is the keyword that begins the image file (the 8 first bytes) and that allows identifying and decoding the image metadata with the 'DREV98' subroutine library version '-a'.

The development of this image header was an experiment to see how the image metadata could be encapsulated in a dynamic way to exchange a maximum of information between an auto-documented sensor and a receiving image database. With this project, it was not our intent to be in competition with the existing COTS image formats such as GIF, PCI, TIFF and NITF (Refs. 1, 2, 3), but rather simply to be able to package the information we wanted to include in an image file, which is not currently allowed by using available COTS image formats.

From this experiment, a number of interesting concepts have emerged. The most important concepts that are worth mentioning are listed below:

1 -The header identifies the numeric representation used for the metadata encoding: then the numbers (integer, float or anything else) can be read, no matter the original processor used to encode the image (Intel processors are 'big endian' (most significant byte right), while the other processors are usually 'little endian' (most significant byte left).

2 -The header length is variable and as many secondary segments as necessary can be stringed together.

3 - The needs for a number of secondary header segments have been identified, developed and tested.

4- The secondary header segments indicate themselves which subroutines must be used to decode them.

5 -The secondary header segments are self-documented and it is possible to decode an unknown segment encoded by a third party, even if this segment has never been seen before. This last point is the most interesting one because it raises the concept of

meta-metadata, i.e. a piece of code that can be read inside the image header that allows us to decode the following part of the header (metadata) .

The information presented in this technical note concurs exactly with the image format developed. The subroutine libraries were developed into two versions, one in FORTRAN and the other in C. Both libraries were tested on a SUN workstation and the C version was also tested on a PC computer. If a new version of this image format needs to be produced, some recommendations are presented in Chapter 5.

This technical note only documents what has been done about the experimental DREV98 image format. However, if it is required that a deliverable product be developed, the recommendations in Chapter 5 will have to be included. This has not been done yet because it was not our intent to develop ready-to-use software (this was in fact part of a larger experiment) and no more resources have been allocated to this sub-project. Moreover, it was concidered better to use a recognized image format like NITF (ref. 3) to keep on with our experiments. Nevertheless, some good ideas have emerged and justify the publication of this note. These ideas can even be communicated to the persons in charge of the recognized image formats (like NITF) to have them create a new version that could incorporate new information encoding concepts (particularly Sections 4.3 and 5.3).

This work was performed at DREV between December 1997 and September 1998 under PSC 5EB12, "Space-Based IR Surveillance".

2.0 THE HEADER STRUCTURE

The structure of the image file is presented in Fig. 1. This image file is composed of a fixed size main header (344 bytes), a series of concatenated secondary header segments and finally the image data.

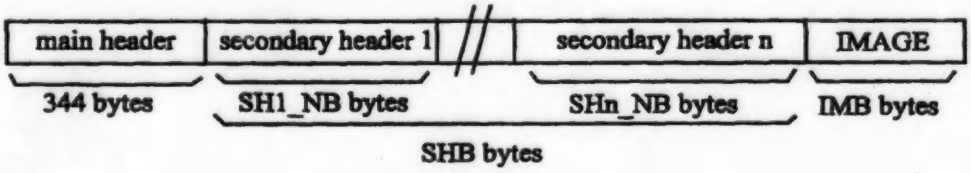

FIGURE 1 - Structure of the image file

The total length of the secondary header segments is recorded in the variable SHB and stored in the main header (byte 341 to 344). Similarly, the total length of the image is recorded in the variable IMB and stored in the main header (byte 337 to 340). Therefore, it is easy to access directly the image by skipping the secondary header.

The secondary header is composed of a multitude of concatenated segments of various formats. It is very easy to navigate in the secondary header because the segment are composed by following strict rules described in Chapter 4. For once, let us say that each segment begins with an integer number (four bytes) that indicates its length. Therefore, it is easy to skip from a segment to another one just by skipping this number of bytes.

3.0 THE MAIN HEADER

The main header is a fixed length structure. It contains all the information (presented in Table I) necessary to decode the secondary header segments and the image data. The definition of the variables follows the table.

The first string that must be read in a DREV98 image file is the first eight bytes to recognize the header format. If the test is successful, the 344 first bytes are read to obtain the complete main header block. Then the variable LITTLE_OR_BIG_ENDIAN indicates how the number must be read. In fact, this variable set a flag in the header decoding method and depending on the encoded file and the kind of computer, the conversion of the numerical values are done appropriately (byte swapping).

The other information contained in the main header (MHD) is the original file name and creation date, image dimensions and numerical representation and total number of bytes of the secondary headers and image. Now, if the user wishes to read the image without decoding the secondary header segments, he can skip directly to the image data.

The content (and structure) of the MHD is presented in Table I, while the description of the variables is presented in Table II. The data types supported by the DREV98 image format are listed in Table III.

TABLE I

Content of the main header

Byte number	Number of bytes	Type	Variable name
1- 8	8	char[8]	HEADER_ID
9 - 264	256	char[256]	FILE_NAME
265 - 296	32	char[32]	CREATION_DATE
297 - 300	4	long	IMAGE_SAMPLE
301 - 304	4	long	IMAGE_LINE
305 - 308	4	long	IMAGE_BAND
309 - 312	4	long	IMAGE_FRAME
313 - 316	4	long	IMAGE_CAMERA
317 - 320	4	long	IMAGE_SPECIAL
321 - 324	4	char[4]	PIXEL_ENCODING_TYPE
325 - 328	4	char[4]	LITTLE_OR_BIG_ENDIAN
329 - 336	8	char[8]	COMPRESSION_TYPE
337 - 340	4	long	IMAGE_BYTES
341 - 344	4	long	SECONDARY_HEADER_BYTES (SHB)

TABLE II

Description of the variable of the main header

Variable	Description
HEADER_ID	Contain the magic character string which allows to identify this header. This character string is, for the current version; 'DREV98-a'.
FILE_NAME	This field contains the name of the file when it was created. Even if the file has been moved, this field still contain the original name.
CREATION_DATE	Creation date of this file
IMAGE_SAMPLE	Number of samples per image line
IMAGE_LINE	Number of lines per image frame
IMAGE_BAND	Number of image bands
IMAGE_FRAME	Number of frames in the image sequence
IMAGE_CAMERA	Number of views or cameras in this archive, e.g., stereoscopic image pairs require to have a left and a right camera, thus 2 cameras.
IMAGE_SPECIAL	Special dimensions for a special application, for example for images acquired with different polarization angles. Hence, with these six dimensions, it is possible to record a sequence of stereoscopic color images acquired in polarized light.
PIXEL_ENCODING_TYPE	Numeric data type of the image (see Table III).
LITTLE_OR_BIG_ENDIAN	This variable is 'L...' for little endian or 'B...' for big endian. Little endian (or less significant byte right) is used by CPUs like Motorola (series 68000), SUN (Spark), while big endian (or most significant byte right) is used by IBM PCs CPUs like the x86 series and Pentium (Intel processors).
COMPRESSION_TYPE	Identification of the algorithm used to compress the image.
IMAGE_BYTES (IMB)	Number of bytes of the image. If no compression is used, then this size is: IMAGE_BYTES = ([SAMPLE * LINE * BAND * FRAM * CAMERA * SPECIAL * nbbitsof (PIXEL_ENCODING_TYPE)] +7) / 8 If a compression is used, then the appropriate method must be called (no compression method is implemented in version DREV98-a).
SECONDARY_HEADER_ BYTES (SHB)	Total number of bytes used by the secondary header segments

The function named "nbbitsof" (used in the calculation of the image size) is a function that returns the number of bits associated to the code contained in the variable PIXEL_ENCODING_TYPE, e.g. "nbbitsof('ui8')" returns the number 64.

TABLE III

Data type supported by the DREV98 image format

Type	Number of bits per pixel	Description
l	1	logical*1 1 bit graphic
2	2	logical*2 2 bit graphic
4	4	logical*4 4 bit graphic
12	12	unsigned 12 bit integer (0 to 1023)
b	8	byte (-128 to 127)
c	8	character
ub	8	unsigned byte or char (0 to 255)
i2	16	short integer
ui2	16	unsigned short integer
i4	32	long integer
ui4	32	unsigned long integer
i8	64	long long integer
ui8	64	unsigned long long integer
f	32	float
d	64	double
ld	128	long double
c	64	complex
dc	128	double complex
rgb	24	24 bits RGB: 8 bits RED, 8 bits GREEN, 8 bits BLUE
rgbt	32	32 bits RGB with transparency: 8 bits RED, 8 bits GREEN, 8 bits BLUE, 8 bits transparency.

4.0 THE SECONDARY HEADER SEGMENTS

While the main header is a fix-block-size data that allows begin decoding of the file, the secondary header is a variable length structure composed of many segments concatenated together. Here is the description of the composition of the segments.

4.1 Why Adding Secondary Header Segments

There are very complex and very complete COTS image formats. For example, TIFF is very good standard extensively used. However, what happens when one needs to include a new kind of information in an image file? He needs to create a new image format. Hence, TIFF cannot record the geographic reference but GE0-TIFF can.

The PCI image file format has the capability of adding secondary header segments to its files, which is already better than the TIFF format. However, the nature of these segments is pre-defined and it is not possible to create new specialized segments, except if you agree that no one else can decode them or simply encode you specific information with ASCII characters, which is a big constraint. Nevertheless, with PCI, it is not necessary to develop a new image format every time a new information must be saved.

4.2 The Generalization of the Header Segments

What is needed is a convention that allows to create an image format convenient enough to include custom information and wise enough to decode it without a priori knowledge provided by the creator. This is not really feasible, except that the required a priori knowledge can be encapsulated by the creator inside the segment itself. Thus, a convention for the encoding and decoding of this a priori knowledge is required and this idea is supported by the DREV98 format. Hence, it will be possible to create new segment types without having to update the decoding library. Unfortunately, this concept has not been exploited to its limits and some recommendations are made in the next Chapter.

4.3 Description of the Structure of the DREV98 Secondary Header Segments

The secondary header segment (SH), presented in Fig. 2, can be used to store any kind of information relevant to the image. It is composed of a fixed length segment header, a variable-size segment descriptor and a variable-size storage area. The description of the segment header variables is presented in Table IV.

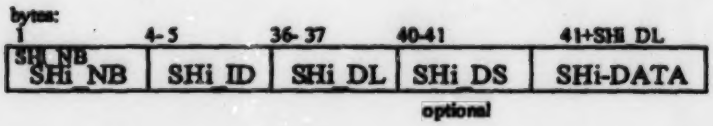

FIGURE 2 - Structure of the variable-length secondary header segment

Since the secondary header is composed of many segment, the variables, in the following description, are indexed par the letter 'i'. Hence, the variable SHi_NB represents the number of bytes (SH_NB) of the i^{th} segment.

The first information encountered in this segment is its length (SHi_NB). Hence, if one does not want to decode this segment, he just has to skip this number of bytes to reach the next segment. This task is performed by a function called 'SHD_GET_NEXT' from the experimental library that was developed during this experiment. This particularity allows to string the segments and access them as if the secondary header were a sequential magnetic tape.

The second information is the identification of the segment (SHi_HD). Thus, by knowing the segment name, it is now possible to call the subroutine (or method) that will be able to decode it. Here, it is strongly recommended to identify the segment with the same name as that of the subroutine used to decode it. An example is presented in Table IV. Also, in the experimental library, a useful function was developed to have the next occurrence of a segment identified by its name. Hence, for example, one can get directly

the 'GEOREFERENCES' segment without having to decode the entire SHD but only by calling a function like "SHD_SEAK ('GEOREFERENCES')".

The next information stored in the segment header is the segment-format descriptor. This descriptor has two fields; an integer (SHi_DL), which indicates the length of the character string, and the character string (SHi_DS) which contains the format of the data saved in this segment. In the current version of the DREV98 file format, the content of this string is more a comment (ASCII syntax) that can be read by the programmer (who can write a new segment decoder when this segment is unknown) than a coded instruction auto-interpreted by a generalized segment decoder. More details are discussed in the next Chapter.

Finally, the last information stored in the segment is an array (SHi_DATA) which contains the image metadata for which this segment is design. The length of this buffer is the total length of the segment, less the length (SHi_DL) of the description string SHi_DS, less the length of the fixed part the segment header (40 bytes), i.e.

Data length (in bytes) = SHi_NB - SHi_DL - 40.

This array can contain almost everything. For example, with the following example:

SH_NB = 89 + 12 * N bytes.
SH_ID = 'SENSOR_CAMERA_SETUP'
SH_DL = 41
SH_DS = 'long N, 1(float), N(long[1]), N(float[2])'
SH_DATA = ... the (12 * N) + 8 following bytes.

the data buffer contains one integer called 'N' (4 bytes), one float (4 bytes), an integer vector (whose length is determined by the content of the integer 'N') and a (two columns by 'N' row) float matrix.

TABLE IV

Description of the variable of the header part of the secondary header segment

SHi_NB	long	Secondary Header _ Number of Bytes: Total length (in bytes) of this segment.
SHi_ID	character*32	Secondary Header _ Identification: String which identifies this data segment. It is strongly recommended that this name matches the name of the routine used to decode this segment. e.g.: SH_ID = 'GEO_REFERENCES' Related routines: GEO_REFERENCES_CREATE GEO_REFERENCES_TRANSLATE GEO_REFERENCES_GET GEO_REFERENCES_PRINT
SHi_DL	long	Secondary Header _ Description Length: SHi_DL = 0 if the field SHi_DS is not used. Length of the character string which describes the data in The SHi_DATA field.
SHi_DS	char*SHi_DL	Secondary Header _ Description String (meta-metadata) String that may contain the formats description and maybe the data titles. This is particularly useful for new secondary header segments developed by new users and which are unknown by the original DREV98-a software package. Hence, the information required to decode foreign header segments may be provided inside this foreign segment itselfs. e.g.: (for the segment GEOREFERENCES) SH_DL = 32 SH_DS ='long N, N(long[9]), N(double[8])' This segment begins with a long integer 'N', followed by N vector of 9 long elements and followed by N vector of 8 double elements.
SHi_DATA	–	Specific to each header segments The length of data is (SHi_NB - SHi_DL - 40) bytes

5.0 RECOMMENDATIONS FOR THE NEXT VERSION

The experimentation with the current version has shown some limitations and bugs. Here are the modifications that will be included in the next version of this image format.

5.1 Bug in Fortran Encoding to C Decoding

The libraries of functions (for the header coding, decoding and manipulation) have been encoded in parallel in C and in Fortran languages. Both libraries work very well independently of the computer platform (tested on PC and SUN workstations). However, a bug was found between the headers encoded with the Fortran library and decoded by the C library. The C functions expect that the character strings be terminated by a null character ('\0'), what the Fortran version is not. Thus, it must be specified as a standard that every character strings be explicitly ended by a '\0', whatever the compiler used to develop the library, i.e. '\0' becomes an element of the header definition and not just a particularity of certain compilers.

5.2 Identification of the Segments

In the current version, a 32-character string is allocated to the variable SHi_ID. In some cases this may not be enough. A 64 (or even a 128) bytes buffer would be better. A variable-length string could be even better and done by adding another parameter to the segment header, like "SHi_IDL" (Secondary Header Identification Length) before the parameter "SHi_ID".

5.3 Segment Format Descriptor

In the example presented in the previous chapter, i.e.:

```
SH_DL  = 41
SH_DS  = 'long N, 1(float), N(long[1]), N(float[2])'
```

SH_DS contains a string (of 41 characters) readable by the programmer. If this segment were unknown, he could write a new subroutine to decode the information contained in this segment. However, this syntax (in a 'C' like style) is not formal enough to be automatically decoded by a generalized segment decoder. A formal syntax should be defined and a parser developed to decode it. This syntax should:

- contain local integer variable for variable-length array description (e.g.: $N, $M)
- support numeric representation presented in Table III (e.g.: i4, f, c, etc.),
- support vector and array definition using parentheses (e.g. f(N,2)),
- support repeating structure indicators (e.g. 2f(n,2)).

Hence, the previous example could become:

SH_DL = 17
SH_DS = '$N,f,i4(N),f(N,2)'

In this example, $N means that an integer (4 bytes) must be read and assigned to the variable 'N' and the value of this variable is reused in the definition of the arrays, which are an integer*4 vector of N elements and a (N,2) float matrix.

With such a syntax, the programmer does not need anymore to write ad-hoc functions because a single all-purpose function could be used to decode every unknown segment. Furthermore, the unknown segments could also contains strings (also self decoded, e.g., 'c(28)' for a vector of 28 characters) that document the data they contain (like a table title).

6.0 CONCLUSION

The image file format presented here was very useful to perform a number experiments on the communication of information between a sensing system and database with a capability of automatic information loading. The development of software related to this image file format has not been fully completed, but this study contains ideas that worth being written.

The file format developed here was useful to communicate information on the sensor setup to the database (optics setup, navigation parameters, etc.), which could not be done with other formats. However, rather than keeping on with the development of an ad-hoc image format, it has been decided to add ad-hoc header segment to other COTS image formats or, even better, to influence the development of these COTS formats.

The concept of self-documented secondary-header segment for the automatic decoding of unknown segments is something that has never been seen in any image format description. This concept of meta-meta-data (i.e. the data that describes the data that documents the data) needs to be further developed and proposed to a recognize image format like NITF for integration. If applied, only a parser is required to decode the image metadata and no more new version of image format will be necessary. It will be possible to include new classes of data inside the image header (without defining a new version of file format) and the end user will still be able to decode this new class of information without updating the image decoding software.

7.0 REFERENCES

1- "PCIDSK C Toolbox", PCI, 50 Weat Wilmot St., Richmond Hill, Ontario, Canada, L4B 1M5, 25 October 1994.

2. - "TIFF Revision 6.0", Adobe Systems Inc., 1585 Charleston Road, P.O. Box 7900, Mountain View, CA 94039-7900, US, June 3 1992.
(http://www.adobe.com/Support/Technotes.html)

3- "National Imagery Transmission Format Standard (NITFS version 2.1)", DOD, MIL-STD-2500A, 12 October 1994.

APPENDIX A

This appendix contains examples (with simulated data) of the use of the header DREV98 and the seven secondary header segments that have already been encoded. These segments are called;

COMPONENT_IDENTIFICATION,
SENSOR_ACQUISITION_TIME,
SENSOR_CAMERA_SETUP,
SENSOR_BAND_DEFINITION,
SENSOR_FLIGHT,
SENSOR_CAMERA_POINTING and
GEO_REFERENCES.

Moreover, they can be used more than once in the secondary header. For example, the segment 'COMPONENT_IDENTIFICATION' could be used five times to identify 1) the camera, 2) the optics, 3) a filter, 4) the carrying platform and 5) the pointing system.

A.1 Example of Printing of Information Contained in the Main Header

The following example shows the metadata saved in the image main header. One can see the file name, creation date, the image dimensions, the pixel definition, the image size and the size of the secondary header. The decoding of this secondary header is shown in the next section.

```
Content of the main image header (DREV98-a):

  - Original file name :  test_01
  - Creation date :       Mon Feb 09 13:47:52 1998

  - Image format:
              SAMPLE   =   512
              LINE     =   512
              BAND     =   3
              FRAME    =   67
              CAMERA   =   1
              SPECIAL  =   1

  - Pixel format:
    PIXEL_ENCODING_TYPE = *i4  * i.e.: integer*4 or long (4 bytes)
    Endian :             B: big endian

  - Compression type :    none

  - Number of image bytes :  210763776

  - Number of bytes in the secondary header:   2807
```

A.2 Secondary-Header Segment: COMPONENT_IDENTIFICATION

This segment contains the identification of a specific component. This component can be for example: CAMERA, POINTING_SYSTEM, PLATFORM, OPTICS, FILTER, and others ...

Segment Description:

Byte	Type	Description
1-4	long	SH_NB = 178
5-36	char*32	SH_ID = 'COMPONENT_IDENTIFICATION'
37-40	long	SH_DL = 10
41-50	char*10	SH_DS = '4(char*32)'
---	---	---
51- 82	char*32	COMPONENT_TYPE
83-114	char*32	COMPONENT_NAME
115-146	char*32	SERIAL_NUMBER
147-178	char*32	COMPANY

Example of Printing Result: (Simulated Data)

```
--------------------------------------------------------------------------------
*  Segment index =  1, segment length =  178 bytes.

  Component identification:
        Component :    Camera
        Model:         K8900
        Serial number: 234E56Y
        Company   :    Kodak

--------------------------------------------------------------------------------
```

*: - The segment index is the byte number (in the secondary header string) where this segment begins (1 in this example), but it can be anything else, see the following examples..
- The segment length is the value indicated by the variable SH_NB.

A.3 Secondary-Header Segment: SENSOR_ACQUISITION_TIME

This segments contains the information about date and time of acquisition

Segment Description:

Byte	Type	Description
1- 4	long	SH_NB = 72
5-36	char*32	SH_ID = 'SENSOR_ACQUISITION_TIME'
37-40	long	SH_DL = 8
41-48	char*8	SH_DS = '6(float)'
---	---	---
49-52	float	START_TIME_YEAR
53-56	float	START_TIME_MONTH
57-60	float	START_TIME_DAY
61-64	float	START_TIME_HOUR
65-68	float	START_TIME_MINUTE
69-72	float	START_TIME_SECOND

Example of Printing Result: (Simulated Data)

```
--------------------------------------------------------------------------------
  Segment index =  179, segment length =  72 bytes.

  Sensor acquisition time:
         Year:          1998.
         Month:         1.
         Day:           12.
         Hour:          11.
         Minute:        29.
         Second:        30.0000

--------------------------------------------------------------------------------
```

A.4 Secondary-Header Segment: SENSOR_CAMERA_SETUP

This segment contains the initial setup of the camera plus a list of parameters (indexed by the frame number), which can change in time.

Segment Description:

Byte	Type	Description
1- 4	long	SH_NB = 93 + 12 * NB_OF_REFERENCED_FRAME bytes.
5-36	char*32	SH_ID = SENSOR_CAMERA_SETUP
37-40	long	SH_DL = 41
41-81	char*41	SH_DS = 'long N, 1(float), N(long[1]), N(float[2])'
—	—	—
82-85	long	NB_OF_REFERENCED_FRAME
86-89	float	CAMERA_FRAME_RATE
90- 93	float	PIXEL_INTEGRATION_TIME
94 - -	long[][1]	FRAME_INDEX
	float[][2]	CAMERA_SETUP

Table Definitions:

long FRAME_INDEX [1] : frame number where the following data is applicable.

float CAMERA_SETUP [] [2] :
[0] : FOCAL_LENGTH
[1] : APERTURE (diaphragm)

Example of Printing Result: (Simulated Data)

```
Segment index =  251, segment length =  189 bytes.

Camera Setup:
       Number of referenced frames:      8
       Frame rate:                      30.00 frames per second.
       Pixel integration time:           0.0170 second.

       Setup:
       Frame        Focal        Aperture
       index        length
         -            (m)          (m)
           1        0.1350       0.0700
           4        0.1350       0.0700
           6        0.1350       0.0600
           9        0.1400       0.0550
          12        0.1403       0.0510
          14        0.1456       0.0480
          19        0.1487       0.0475
          32        0.1500       0.0467
```

A.5 Secondary-Header Segment: SENSOR_BAND_DEFINITION

This segment contains the information that describes the wavelength limits of the wavebands specified by the band index.

Segment Description:

Byte	Type	Description
1- 4	long	SH_NB= 75 + 28 * NB_OF_DESCRIBE_BAND bytes.
5- 36	char*32	SH_ID = 'SENSOR_BAND_DEFINITION'
37-40	long	SH_DL = 31
41-71	char*31	SH_DS = 'long N, N(long[1]), N(float[6])'
—	—	—
72-75	long	NB_OF_DESCRIBE_BAND
76- -	long[] [1]	BAND_INDEX
- - -	float[] [6]	WAVEBAND_DESCRIPTION

Table Definitions:

long BAND_INDEX [] [1] : Band number where the following data is applicable

float WAVEBAND_DESCRIPTION [] [2] :
[0] : WAVELENGTH_MINIMUM
[1] : WAVELENGTH_MAXIMUM
[2] : FIRST_HYPERSPECTRAL_CHANNEL
[3] : LAST_HYPERSPECTRAL_CHANNEL
[4] : BAND_ACQUISITION_TIME *
[5] : PIXEL_INTEGRATION_TIME *
*: different for image scanner, equal for mosaic detector.

Example of Printing Result: (Simulated Data)

```
Segment index =  440, segment length =  111 bytes.

Band definition:
      Number of referenced bands:     3

  Band     Minimum      Maximum      First     Last     Acq.     Pixel acq.
 index   wavelength   wavelength   channel   channel   time       time
   -        (um)         (um)                          (s)        (s)
   1       3.000        4.200        1         3       0.021    0.00018
   2       4.200        4.400        4         4       0.021    0.00018
   3       4.400        5.100        5         7       0.021    0.00018
```

A.6 Secondary-Header Segment: SENSOR_FLIGHT

This segment contains the information that describes the sensor attitude for a number of frames (specified by the frame index) of the image sequence.

Segment Description:

Byte	Type	Description
1- 4	long	SH_NB= 77 + 100 * NB_OF_REFERENCED_FRAME bytes .
5- 36	char*32	SH_ID = 'SENSOR_FLIGHT'
37-40	long	SH_DL = 33
41-73	char*33	SH_DS = 'long N, N(long[1]), N(double[12])'
—	—	—
74-77	long	NB_OF_REFERENCED_FRAME
78- -	long[] [1]	FRAME_INDEX
- - -	double[][12]	SENSOR_ATTITUDE

Table Definitions:

long FRAME_INDEX [][1] : frame numbers where the following data is applicable.

double SENSOR_ATTITUDE [][12] :
[0] SENSOR_LONGITUDE
[1] SENSOR_LATITUDE
[2] SENSOR_ALTITUDE
[3] SENSOR_HEADING
[4] SENSOR_SPEED
[5] SENSOR_VERTICAL_SPEED
[6] SENSOR_PITCH
[7] SENSOR_YAW
[8] SENSOR_ROLL
[9] SENSOR_PITCH_RATE
[10] SENSOR_YAW_RATE
[11] SENSOR_ROOL_RATE

Example of Printing Result: (Simulated Data)

Segment index = 551, segment length = 877 bytes.

Sensor flight:
Number of referenced frames: 8

Frame	Longitude (deg)	Latitude (deg)	Altitude (m)	Heading (deg)	Horizontal speed (m/s)	Vertical speed (m/s)
	Pitch (deg)	Yaw (deg)	Roll (deg)	dPitch/dt (deg/s)	dYaw/dt (deg/s)	dRoll/dt (deg/s)
1	12.12345695	45.11678838	1000.4	180.0	200.0	12.4
	0.0	0.1	0.0	1.00	2.00	1.00
4	12.12345695	45.09678838	1001.6	180.2	200.0	12.7
	0.1	0.3	0.1	1.00	2.00	1.00
6	12.12345695	45.08345505	1002.5	180.4	200.0	13.0
	0.2	0.4	0.2	1.00	2.00	1.00
9	12.12345695	45.06345505	1003.7	180.7	200.0	13.3
	0.3	0.6	0.3	1.00	2.00	1.00
12	12.12345695	45.04345505	1004.9	181.1	200.0	13.6
	0.4	0.8	0.4	1.00	2.00	1.00
14	12.12345695	45.03012171	1005.7	181.5	200.0	13.9
	0.5	0.9	0.5	1.00	2.00	1.00
19	12.12345695	44.99678838	1007.8	182.2	200.0	14.4
	0.6	1.3	0.6	1.00	2.00	1.00
32	12.12345695	44.91012170	1013.1	183.2	200.0	15.9
	1.1	2.1	1.1	1.00	2.00	1.00

A.7 Secondary-Header Segment: SENSOR_CAMERA_POINTING

This segment contains the camera pointing parameters which can change in time (indexed by the frame number).

Segment Description:

Byte	Type	Description
1-4	long	SH_NB = 76 + 52 * NB_OF_REFERENCED_FRAME bytes.
5-36	char*32	SH_ID = SENSOR_CAMERA_POINTING
37-40	long	SH_DL = 32
41-72	char*32	SH_DS = 'long N, N(long[1]), N(double[6])'
—	—	—
73-76	long	NB_OF_REFERENCED_FRAME
77-80	LONG	ABSOLUTE_POINTING (1:absolute in the Earth frame of reference, 0: relative, in the airframe frame of reference)
81 - -	long[][1]	FRAME_INDEX
- - -	double[] [6]	CAMERA_POINTING

Table Definitions:

long FRAME_INDEX [][1] : frame number where the following data is applicable.

double CAMERA_POINTING [] [6] :

[0] AZIMUTHAL_ANGLE
[1] ELEVATION_ANGLE
[2] ROTATION_ANGLE
[3] AZIMUTHAL_ANGLE_RATE
[4] ELEVATION_ANGLE_RATE
[5] ROTATION_ANGLE_RATE

Example of Printing Result: (Simulated Data)

Segment index = 1428, segment length = 492 bytes.

Camera pointing:
Number of referenced frames: 8

Frame	Azimuth angle (deg)	Elevation angle (deg)	Rotation angle (deg)	Azimuth spin (deg/s)	Elevation spin (deg/s)	Rotation spin (deg/s)
1	90.50	-44.90	0.60	15.00	3.00	12.00
4	92.00	-44.60	1.80	15.00	3.00	12.00
6	93.00	-44.40	2.60	15.00	3.00	12.00
9	94.50	-44.10	3.80	15.00	3.00	12.00
12	96.00	-43.80	5.00	15.00	3.00	12.00
14	97.00	-43.60	5.80	15.00	3.00	12.00
19	99.50	-43.10	7.80	15.00	3.00	12.00
32	106.00	-41.80	13.00	15.00	3.00	12.00

A.8 Secondary-header segment: GEOREFERENCES

This segment contains the information required to geo-reference an image. It contains a set of longitude and latitude for the TOP, BOTTOM, LEFT and RIGHT pixels of the image corners or other pixel, as specified by the table REFERENCED_PIXEL. With this approach, the geo-references are not necessarily applied to the real image corners but rather with the pointed pixel. For example, in the case where the horizon is included in the image, the TOP_LEFT pixel (which is usually the image corner with the index [0][0] in C or (1,1) in Fortran) can be replaced by a pixel located below the horizon, otherwise, it would be impossible to geo-reference a pixel that shows the sky. Also, this segment can contain many sets of geo-references, each set being associated with a specific frame of the image sequence.

Segment Description:

Byte	Type	Description
1- 4	long	SH_NB = 88 + 100 * NB_OF_REFERENCED_FRAME bytes.
5- 36	char*32	SH_ID = 'GEOREFERENCES'
37-40	long	SH_DL = 44
41-84	char*44	SH_DS = 'long N, N(long[1]), N(long[8]), N(double[8])'
—	—	—
85-88	long	NB_OF_REFERENCED_FRAME
89- -	long[] [1]	FRAME_INDEX
- - -	long[] [8]	REFERENCED_PIXEL
- - -	double [] [8]	GEOREFERENCES

Table Definitions:

long FRAME_INDEX [][1]

long REFERENCED_PIXEL [] [8]:
- [0] TOP_LEFT_SAMPLE
- [1] TOP_LEFT_LINE
- [2] TOP_RIGHT_SAMPLE
- [3] TOP_RIGHT_LINE
- [4] BOTTOM_LEFT_SAMPLE
- [5] BOTTOM_LEFT_LINE
- [6] BOTTOM_RIGHT_SAMPLE
- [7] BOTTOM_RIGHT_LINE

double GEO_REFERENCES [] [8]:
- [0] TOP_LEFT_LONGITUDE
- [1] TOP_LEFT_LATITUDE
- [2] TOP_RIGHT_LONGITUDE
- [3] TOP_RIGHT_LATITUDE
- [4] BOTTOM_LEFT_LONGITUDE
- [5] BOTTOM_LEFT_LATITUDE
- [6] BOTTOM_RIGHT_LONGITUDE
- [7] BOTTOM_RIGHT_LATITUDE

Example of Printing Result: (Simulated Data)

Segment index = 1920, segment length = 888 bytes.

Geo-references:

Number of referenced frames: 8

Frame	Corner	--- Pixel --- Sample	Line	------ Geo-references ------ Longitude (deg)	Latitude (deg)
1	TL	1	1	100.10000000	45.00000000
	TR	1	512	98.00000000	45.00000000
	BL	512	1	100.10000000	44.00000000
	BR	512	512	98.00000000	44.00000000
4	TL	1	1	100.10000000	45.00000000
	TR	1	512	98.00000000	45.00000000
	BL	512	1	100.10000000	44.00000000
	BR	512	512	98.00000000	44.00000000
6	TL	1	1	100.10000000	45.00000000
	TR	1	512	98.00000000	45.00000000
	BL	512	1	100.10000000	44.00000000
	BR	512	512	98.00000000	44.00000000
9	TL	1	1	100.10000000	45.00000000
	TR	1	512	98.00000000	45.00000000
	BL	512	1	100.10000000	44.00000000
	BR	512	512	98.00000000	44.00000000
12	TL	1	1	100.10000000	45.00000000
	TR	1	512	98.00000000	45.00000000
	BL	512	1	100.10000000	44.00000000
	BR	512	512	98.00000000	44.00000000
14	TL	1	1	100.10000000	45.00000000
	TR	1	512	98.00000000	45.00000000
	BL	512	1	100.10000000	44.00000000
	BR	512	512	98.00000000	44.00000000
19	TL	1	1	100.10000000	45.00000000
	TR	1	512	98.00000000	45.00000000
	BL	512	1	100.10000000	44.00000000
	BR	512	512	98.00000000	44.00000000
32	TL	1	1	100.10000000	45.00000000
	TR	1	512	98.00000000	45.00000000
	BL	512	1	100.10000000	44.00000000
	BR	512	512	98.00000000	44.00000000

End of secondary header reached, total length = 2807 bytes.

INTERNAL DISTRIBUTION

DREV TN 2000-050

1 - Director General
1 - Deputy Director General
1 - Chief Scientist
6 - Document library
1 - M. P. Lévesque (Author)
1 - T. Smithson
1 - J. M. Garneau
1 - A. Blanchard
1 - J.-P. Ardouin
1 - J. Boiteau
1 - G. Fournier
1 - D. Gouin
1 – A. Beaudouin

EXTERNAL DISTRIBUTION

DREV TN 2000-050

1 - DRDCIM
1 - DRDCIM (unbound copy)

SANS CLASSIFICATION
COTE DE SÉCURITÉ DE LA FORMULE
(plus haut niveau du titre, du résumé ou des mots-clefs)

FICHE DE CONTRÔLE DU DOCUMENT

1. PROVENANCE (le nom et l'adresse) Centre de recherches pour la défense, Valcartier	2. COTE DE SÉCURITÉ (y compris les notices d'avertissement, s'il y a lieu)

3. TITRE (Indiquer la cote de sécurité au moyen de l'abréviation (S, C, R ou U) mise entre parenthèses, immédiatement après le titre.)
Description of the DREV96 Image Format

4. AUTEURS (Nom de famille, prénom et initiales. Indiquer les grades militaires, ex.: Bleau, Maj. Louis E.)
Martin P. Lévesque

5. DATE DE PUBLICATION DU DOCUMENT (mois et année) janvier 2001	6a. NOMBRE DE PAGES 26	6b. NOMBRE DE REFERENCES 3

7. DESCRIPTION DU DOCUMENT (La catégorie du document, par exemple rapport, note technique ou mémorandum. Indiquer les dates lorsque le rapport couvre une période définie.)
Note technique

8. PARRAIN (le nom et l'adresse)
SEB12

9a. NUMÉRO DU PROJET OU DE LA SUBVENTION (Spécifier si c'est un projet ou une subvention)	9b. NUMÉRO DE CONTRAT
10a. NUMÉRO DU DOCUMENT DE L'ORGANISME EXPÉDITEUR	10b. AUTRES NUMÉROS DU DOCUMENT N/A

11. ACCÈS AU DOCUMENT (Toutes les restrictions concernant une diffusion plus ample du document, autres que celles inhérentes à la cote de sécurité.)

- [x] Diffusion illimitée
- [ ] Diffusion limitée aux entrepreneurs des pays suivants (spécifier)
- [ ] Diffusion limitée aux entrepreneurs canadiens (avec une justification)
- [ ] Diffusion limitée aux organismes gouvernementaux (avec une justification)
- [ ] Diffusion limitée aux ministères de la Défense
- [ ] Autres

12. ANNONCE DU DOCUMENT (Toutes les restrictions à l'annonce bibliographique de ce document. Cela correspond, en principe, aux données d'accès au document (11). Lorsqu'une diffusion supplémentaire (à d'autres organismes que ceux précisés à la case 11) est possible, on pourra élargir le cercle de diffusion de l'annonce.)

SANS CLASSIFICATION
COTE DE LA SÉCURITÉ DE LA FORMULE
(plus haut niveau du titre, du résumé ou des mots-clefs)

dcd03f ver.(08-1999)

SANS CLASSIFICATION

COTE DE LA SÉCURITÉ DE LA FORMULE
(plus haut niveau du titre, du résumé ou des mots-clefs)

13. SOMMAIRE (Un résumé clair et concis du document. Les renseignements peuvent aussi figurer ailleurs dans le document. Il est souhaitable que le sommaire des documents classifiés soit non classifié. Il faut inscrire au commencement de chaque paragraphe du sommaire la cote de sécurité applicable aux renseignements qui s'y trouvent, à moins que le document lui-même soit non classifié. Se servir des lettres suivantes: (S), (C), (R) ou (U). Il n'est pas nécessaire de fournir ici des sommaires dans les deux langues officielles à moins que le document soit bilingue.)

The DREV98 image format was developed to conduct a number of experiments on the transfer of metadata between an auto-documented sensor and an auto-loading image database. These experiments allowed us to determine which metadata had to be stored in the image header. Moreover, this image format brings the concept that the header segments should contain meta-metadata, i.e. information that describes the header segment data (which itself describes the image) allowing a universal decoder to decode all unknown segment. This technical note contains the description of the DREV98 image format and the new concept of the self-documented header segments.

14. MOTS-CLÉS, DESCRIPTEURS OU RENSEIGNEMENTS SPÉCIAUX (Expressions ou mots significatifs du point de vue technique, qui caractérisent un document et peuvent aider à le cataloguer. Il faut choisir des termes qui n'exigent pas de cote de sécurité. Des renseignements tels que le modèle de l'équipement, la marque de fabrique, le nom de code du projet militaire, la situation géographique, peuvent servir de mots-clés. Si possible, on doit choisir des mots-clés d'un thésaurus, par exemple le "Thesaurus of Engineering and Scientific Terms (TESTS)". Nommer ce thésaurus. Si l'on ne peut pas trouver de termes non classifiés, il faut indiquer la classification de chaque terme comme on le fait avec le titre.)

image file, image header, file format, header segment, metadata, meta-metadata

515228

SANS CLASSIFICATION

COTE DE SÉCURITÉ DE LA FORMULE
(plus haut niveau du titre, du résumé ou des mots-clefs)